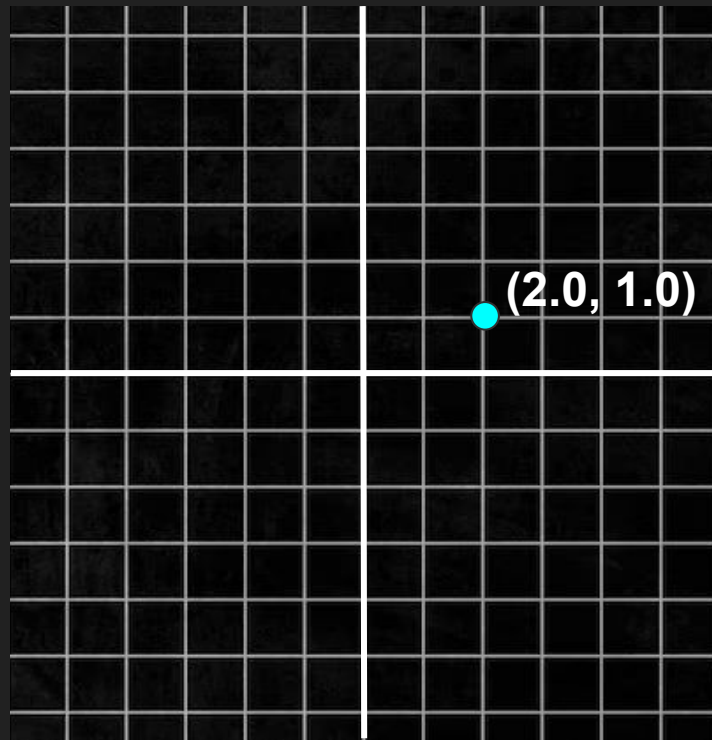# CL28: OOP Practice

Let's use some Point objects to make a Line!

Consider this Point class

```python
class Point:
    x: float
    y: float

    def __init__(self, x: float, y: float):
        self.x = x
        self.y = y

    def dist_from_origin(self) -> float:
        return (self.x**2 + self.y**2) ** 0.5

    def translate_x(self, dx: float) -> None:
        self.x += dx

    def translate_y(self, dy: float) -> None:
        self.y += dy

pt: Point = Point(2.0, 1.0)
```
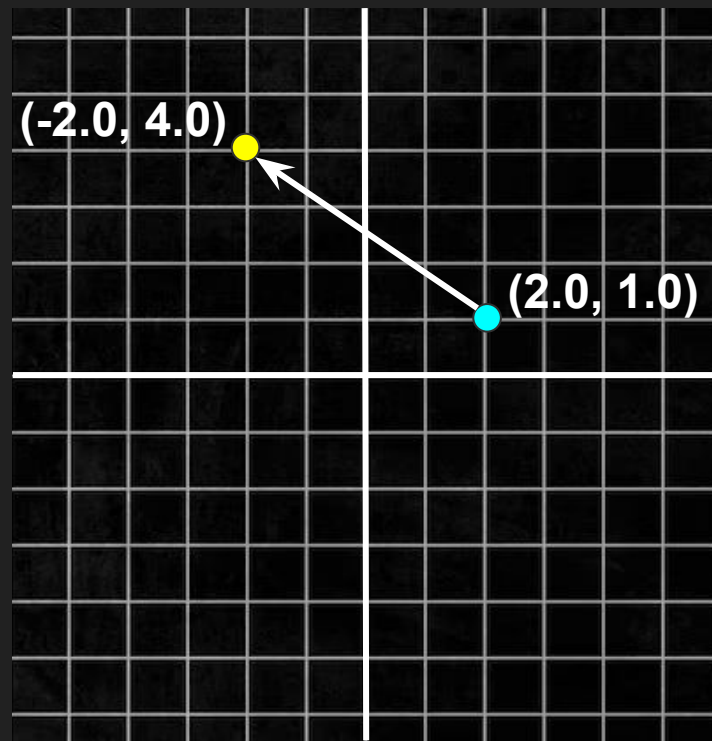
(2.0, 1.0)

"Two points make a line"

Finding the length of a line:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Finding the slope of a line:

$$m = \frac{Rise}{Run} = \frac{y_2 - y_1}{x_2 - x_1}$$

(-2.0, 4.0)

(2.0, 1.0)

"Two points make a line"

Let's define a Line class and use it to see the distance from the Old Well to the Planetarium!

On paper, write a **Line** class with two attributes: a starting point (**start: Point**) and an ending point (**end: Point**).
The **Line** class should have the following method definitions:

- ```
  def __init__(self, start: Point,
  end: Point):
  ```

- ```
  def get_length(self) -> float:
  ```
  - Calculates the length of the line

- ```
  def get_slope(self) -> float:
  ```
  - Calculates the slope (from **start** to **end**)

(7.0, 5.0)

(2.0, 1.0)
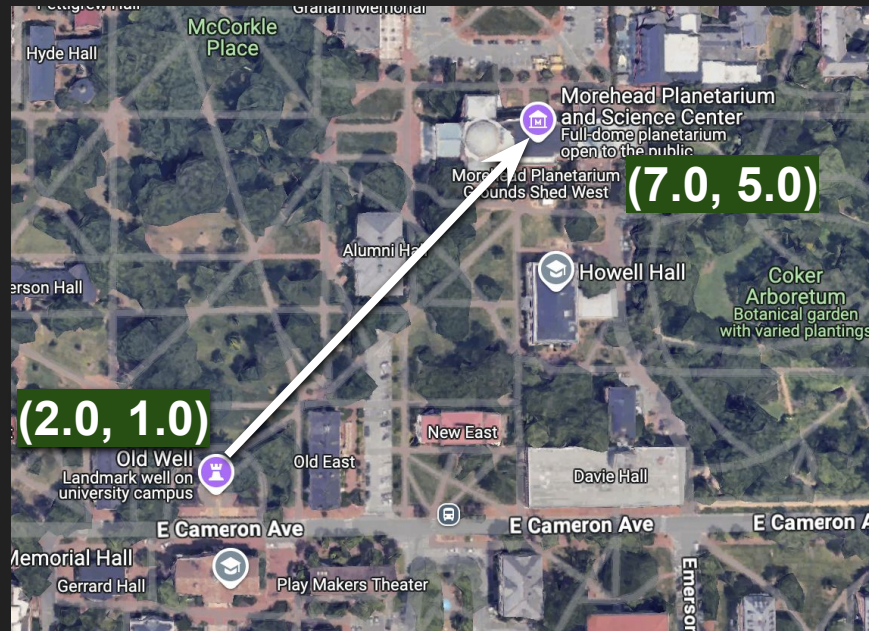
"Two points make a line" – Let's make a Line class!

Finding the length of a line:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Finding the slope of a line:

$$m = \frac{Rise}{Run} = \frac{y_2 - y_1}{x_2 - x_1}$$

Step 1: *On paper*, write a `Line` class with two attributes: a starting point (`start: Point`) and an ending point (`end: Point`).

The `Line` class should have the following method definitions:

- **Step 2:** `def __init__(self, start: Point, end: Point):`

- **Step 3:** `def get_length(self) -> float:` calculates the length of the line

- **Step 4:** `def get_slope(self) -> float:` calculates the slope (from `start` to `end`)

Let's go over it together! →

```
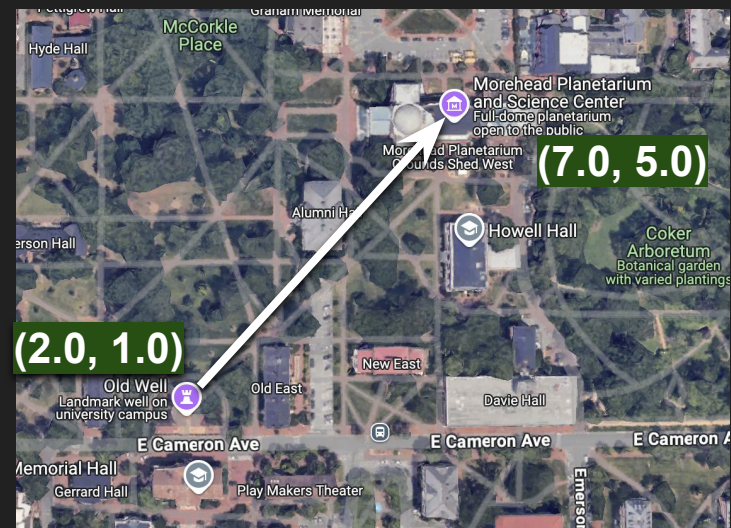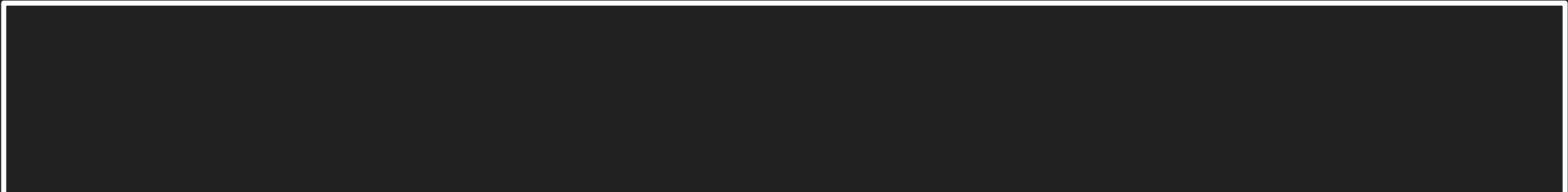1-16 class Point: … # collapsed for space
17
18 class Line:
19     start: Point
20     end: Point
21
22     def __init__(self, start: Point, end: Point):
23         self.start = start
24         self.end = end
25
26     def get_length(self) -> float:
27         x_diffs: float = self.end.x - self.start.x
28         y_diffs: float = self.end.y - self.start.y
29         return (x_diffs**2 + y_diffs**2) ** 0.5
30
31     def get_slope(self) -> float:
32         x_diffs: float = self.end.x - self.start.x
33         y_diffs: float = self.end.y - self.start.y
34         return y_diffs / x_diffs
```


(7.0, 5.0)

(2.0, 1.0)

Create a Line object and find the distance from the Old Well to the Planetarium:

```
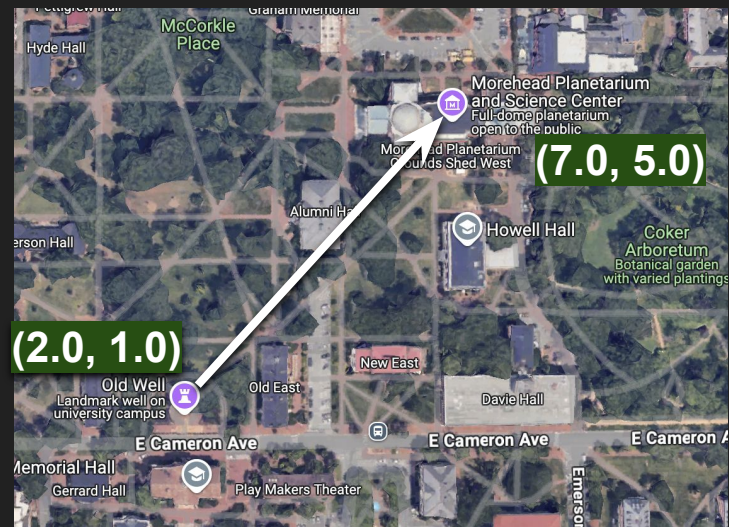1-16 class Point: … # collapsed for space
17
18 class Line:
19     start: Point
20     end: Point
21
22     def __init__(self, start: Point, end: Point):
23         self.start = start
24         self.end = end
25
26     def get_length(self) -> float:
27         x_diffs: float = self.end.x - self.start.x
28         y_diffs: float = self.end.y - self.start.y
29         return (x_diffs**2 + y_diffs**2) ** 0.5
30
31     def get_slope(self) -> float:
32         x_diffs: float = self.end.x - self.start.x
33         y_diffs: float = self.end.y - self.start.y
34         return y_diffs / x_diffs
```


(7.0, 5.0)

(2.0, 1.0)

Create a Line object and find the distance from the Old Well to the Planetarium:

```
me: Point = Point(2.0, 1.0)
planet_y: float = 5.0
planet_loc: Point = Point(me.x + 5, planet_y) # Example of accessing an attribute's value
path: Line = Line(me, planet_loc)
print(path.get_slope())
```

Submit a .pdf of your hand-written code to Gradescope!

Want extra practice? Try diagramming this!

```
1-16  class Point: … # collapsed for space
  17
  18  class Line:
  19      start: Point
  20      end: Point
  21
  22      def __init__(self, start: Point, end: Point):
  23          self.start = start
  24          self.end = end
  25
  26      def get_length(self) -> float:
  27          x_diffs: float = self.end.x - self.start.x
  28          y_diffs: float = self.end.y - self.start.y
  29          return (x_diffs**2 + y_diffs**2) ** 0.5
  30
  31      def get_slope(self) -> float:
  32          x_diffs: float = self.end.x - self.start.x
  33          y_diffs: float = self.end.y - self.start.y
  34          return y_diffs / x_diffs
  35
  36  me: Point = Point(2.0, 1.0)
  37  planet_y: float = 5.0
  38  planet_loc: Point = Point(me.x + 5, planet_y)
  39  path: Line = Line(me, planet_loc)
  40  print(path.get_slope())
```